Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/5540--95-7733

Epistemology of Information Flow in the Multilevel Security of Probabilistic Systems

JAMES W. GRAY, III

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon Hong Kong

PAUL F. SYVERSON

Center for High Assurance Computing Systems
Information Technology Division

May 12, 1995

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	May 12, 1995	Interim

4. TITLE AND SUBTITLE

Epistemology of Information flow in the Multilevel Security of Probabilistic Systems

5. FUNDING NUMBERS

WU-55-2830-0-5
PE-61153N

6. AUTHOR(S)

James W. Gray, III* and Paul F. Syverson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5540–95-7733

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*Department of Computer Science, Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

We set out a modal logic for reasoning about multilevel security of probabilistic systems. This logic includes modalities for time, probability, knowledge, and permitted-knowledge. Making use of the Halpern-Tuttle framework for reasoning about knowledge and probability, we give a semantics for our logic and prove that it is sound. We give two syntactic definitions of perfect multilevel security and show that their semantic interpretations are equivalent to two earlier, independently motivated characterizations. We also discuss the relation between these characterizations of security and between their usefulness in security analysis.

14. SUBJECT TERMS

Covert channels	Multilevel security	Models of computation
Information flow	Specification	Knowledge
Logic	Verification	Knowledge representation

15. NUMBER OF PAGES

48

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

EPISTEMOLOGY OF INFORMATION FLOW IN THE MULTILEVEL SECURITY OF PROBABILISTIC SYSTEMS

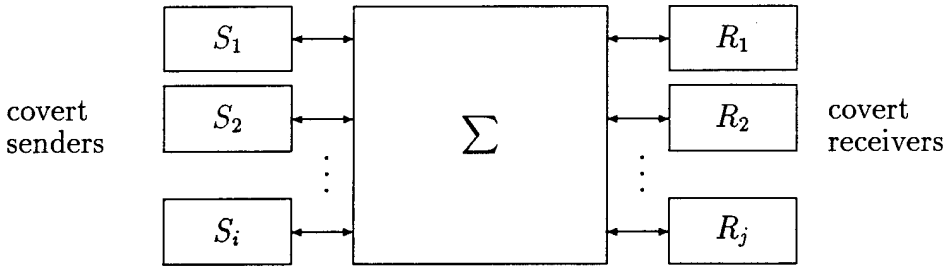

Figure 1: The General Form of a System

1 Introduction

Multilevel security is the aspect of computer security concerned with protecting information that is classified with respect to a multilevel hierarchy (e.g., UNCLASSIFIED, SECRET, TOP SECRET). A *probabilistic system* is a hardware or software system that makes probabilistic choices (e.g., by consulting a random number generator) during its execution. Such probabilistic choices are useful in a multilevel security context for introducing noise to reduce the rate of (or eliminate) illicit communication between processes at different classification levels. In this paper, we are concerned with definitions of *perfect* (information-theoretic) multilevel security in the sense that the definitions rule out *all* illicit communication without relying on any complexity-theoretic assumptions. That is, our model allows the system penetrators to have unlimited computational power and yet, our definitions are still sufficient to ensure that there can be no illicit communication.[1]

The systems that we address can be depicted in the form shown in Figure 1. This general form is intended to represent systems including physical hardware with hard-wired connections to other systems, an operating system kernel with connections to other processes provided by shared memory, and processes executing on a multiprocessor with connections to other systems provided by an interprocess communication (IPC) mechanism.

• There is a system, called Σ, that provides services to the other systems. For example, in the case of a multiuser relational database, Σ would store and control access to a set

[1]Of course in practice we do not have true random number generators—merely pseudo-random number generators—and so, systems that depend on random number generators for their security will not be able to achieve the ideal of perfect multilevel security. In such cases, one would want to prove, e.g., that under the assumption that penetrators are limited to a polynomial amount of time, the pseudo-random number generator is as good as random. We relegate such considerations to a lower level of analysis.

Manuscript approved January 18, 1995.

of relations. Σ is the system with respect to which we will be reasoning about multilevel security.

• There is a set of systems (labeled S_1, S_2, ..., S_i in the figure), called the "covert senders", that have access to secret information. These systems are called "covert senders" because they may attempt to covertly send secret information, via Σ, to other systems that are not authorized to see the information. It is these attempts with which we are concerned. As is commonly done in the literature, we will often refer to the covert senders as *high* systems (referring to the situation where the covert senders have access to *highly classified* information). We will also refer to the set of covert senders collectively as the high *environment*, denoted $\mathcal{H}$. These systems are part of "the environment" in the sense that they are in the environment of the central system, Σ.

• There is a second set of systems (labeled R_1, R_2, ..., R_j in the figure), called the "covert receivers", that are not authorized to see the secret information that is available to the covert senders. We will often refer to the covert receivers as *low* systems, or collectively as the low *environment*, denoted $\mathcal{L}$.

If the covert senders are able to use Σ to communicate information to the covert receivers, we will say that Σ has a *covert channel*, or equivalently (for our purposes) that Σ is *insecure*. A few notes are in order.

1. It is important to bear in mind that the threat that we are concerned with is *not* that the users (i.e., the *human* users) of the covert sender systems are attempting to send secret information to the covert receivers. We assume that if they wanted to, they could more easily pass notes in the park and entirely bypass Σ. Rather, we are concerned that the covert senders are actually trojan horses (i.e., they appear to be something that the user wants, but actually contain something else that is entirely undesirable to the user) and that these trojan horses are attempting to send secret information to the covert receivers. This is a legitimate concern since system developers do not want to incur the cost of verifying every component of a conglomerate system with respect to multilevel security requirements. Ideally, only a small number of components in the system (e.g., in our case only Σ) have security requirements, and so require verification; while the remaining components can be implemented by off-the-shelf hardware and software that are unverified with respect to security (and therefore may be trojan horses).

We assume a worst case scenario, where *all* of the covert senders and covert receivers are trojan horses: Indeed, we assume that *all* of the trojan horses are cooperating in an attempt to transmit information from the covert senders to the covert receivers.

2. It is also important to bear in mind that in our intended application, the covert senders will not be able to communicate directly to the covert receivers (i.e., by bypassing Σ). Typically, there are hardware or software controls to prevent this. For example, non-bypassability is one of the well-known principles of a "reference monitor" (see [Gas88]), which is one of the typical applications we have in mind.

3. Our model contrasts sharply with much other work on security (e.g., [Mea92], [DDWY93]) in that we consider a set of untrusted agents (viz, the covert senders and receivers) that are connected via a trusted agent, whereas these other works consider a set of *trusted* agents connected via an *untrusted* agent. This difference in our model reflects the difference in the respective applications. The work of [Mea92] and [DDWY93] is intended to be used to analyze a set of legitimate (and trusted) agents that are attempting to establish secure communication over an untrusted network. In that work, the assumption is that the penetrator is able to subvert the network (i.e., the central component of the system), but not the trusted (lateral) agents.

   In contrast, our work is intended to be used to analyze a centralized server that serves a set of untrusted entities. Correspondingly, our assumption is that the penetrator may be able to subvert the untrusted (lateral) agents, but not the central server.

4. The fact that we have partitioned the set of systems external to Σ into two sets, high and low, may seem to indicate that we are limiting ourselves to two levels of information (e.g., SECRET and UNCLASSIFIED). However, this is not the case. In a more general setting, information is classified (users are cleared, resp.) according to a finite, partially ordered set (see e.g., [Den76]); that is, there is a finite set of classification levels (clearance levels, resp.) that is ordered by a reflexive, transitive, and anti-symmetric relation, which we call *dominates*. A given user is permitted to observe a given piece of information only if the user's clearance dominates the classification of the information. In the case where there are more than two levels, a separate analysis would be performed for each level, x; in each analysis, the set of levels would be partitioned into those that are dominated by x (i.e., the "low" partition) and the set

of levels that are not dominated by x (i.e., the "high" partition). Thus, we have lost no generality by restricting our attention to two levels.

Our problem is to develop a logic that can be used to reason about the multilevel security of a given system Σ. In particular, we would like to be able to verify whether or not a given (probabilistic or deterministic) system has any covert channels. Our approach is similar to Glasgow, MacEwen, and Panangaden's [GMP90] and Bieber and Cuppens' [BC92] in that our primary definition of security is given in terms of modal logic. In particular, as in [BC92], we say that a system is secure with respect to the set of low processes, denoted L, if and only if for any logical formula φ, the following formula is derivable from the given premises, describing e.g., the behavior of the system Σ.

$$\Box(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)) \tag{1}$$

where $\Box(\psi)$ is intuitively regarded as *always* ψ,[2] $\mathbf{K}_L(\varphi)$ is intuitively regarded as "L *knows* φ" and $R_L(\varphi)$ is intuitively regarded as "L *is permitted to know* φ." Our work extends that of [GMP90] and [BC92] in that our logic includes explicit means to specify and reason about the *probabilistic* behavior of systems. That is, in our logic, the formula φ may say e.g., "the probability of a given high process's input is .99". For such a φ, the formula $\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)$ says that if L *knows* that the probability of a given high process's input is .99, then L is *permitted to know* that the probability of that high process's input is .99.

The motivation for reasoning about the probabilistic behavior of systems has appeared in examples and discussions of many authors (cf. [Bro91, Gra92, MR88, McC88, McL90, WJ90]). Essentially, the motivation is that it is possible for a probabilistic system to satisfy many existing definitions of security (e.g., Sutherland's *Nondeducibility* [Sut86], McCullough's *Restrictiveness* [McC90], etc.) and still contain probabilistic covert channels.

Others have developed logics to reason about knowledge and probability in the areas of artificial intelligence (viz, Ruspini [Rus87]) and protocol analysis (viz, Fagin and Halpern [FH94]). Semantically, the framework of Halpern and Tuttle ([HT93]) encompasses the other two and, in fact, we are also able to make use of their framework to give a semantics to our logic.

A primary contribution of the present paper is the unification of the logical approach to multilevel security developed by Glasgow, MacEwen, and Panangaden [GMP90] and Bieber

[2]For technical reasons, Bieber and Cuppens' definition omitted the $\Box$ operator.

and Cuppens [BC92] with the work on security of probabilistic systems done by McLean [McL90], Browne [Bro89], and the first author [Gra92]. In particular, we prove that the semantic interpretation of (1) is equivalent to Gray's *Probabilistic Noninterference* (which is itself equivalent to Browne's *Stochastic Non-Interference*). We also give a verification condition (in our logic) and prove that it is equivalent to Gray's *Applied Flow Model* (which is closely related to McLean's *Flow Model*). These results are doubly advantageous. On the one hand they constitute a formalization of the just cited information-theoretic approaches to security. On the other hand, to the extent that the just cited logical works are viewed not just as formalizations but as another basic approach to security, the results in this paper amount to a demonstration of the equivalence of independently motivated characterizations of security. We consider this to be strong evidence that both characterizations have 'got things right'. For a discussion of the importance of such equivalences see, e.g., [McL87].

The remainder of the paper is organized as follows. In §2 we set out our model of computation. In §§3 and 4, we set out the syntax and semantics of our logic, and in §5, we prove its soundness. In §6 we state our primary definition of security and prove that it is equivalent to Probabilistic Noninterference. In §7 we state our verification condition and show that it is equivalent to the Applied Flow Model. Finally, in §8, we give some conclusions of this work.

2 System Model

In this section, we describe our system model. This is the model by which we will (in §4) give semantics to our logic. First, we describe the general system model, which is taken from Halpern and Tuttle [HT93]. Then, we will tailor the model to our needs by (in Halpern and Tuttle's terminology) choosing the "adversaries". Finally, we impose some additional structure on the model, resulting in our application-specific model.

2.1 General System Model

In this subsection we review the general system model of Halpern and Tuttle. A complete description of their model can be found in [HT93].

We have a set of agents, $P_1, P_2, \ldots, P_n$, each with its own local state. The *global state* is an n-

tuple of the local agents' states.[3] A *run* of the system is a mapping of times to global states. We assume that time is discrete because we are dealing with security at the digital level of the system. We are not, for example, addressing security issues such as analog channels in hardware. Therefore, as in [HT93], we will assume that times are natural numbers.

The probabilities of moving among global states are represented in the model by means of labeled computation trees. The nodes of the trees represent global states. For any given node in a tree, the children of that node represent the set of global states that could possibly come next. Each arc from a node to one of its children is labeled with the probability of moving to that state. Thus, from any given node, the sum of the probabilites on its outgoing arcs must be one. As in [HT93], we also assume that the set of outgoing arcs is finite and that all arcs are labeled with *nonzero* probabilities. This final assumption can be viewed as a *convention* that if the probability of moving from state x to state y is zero, then state y is not included as a child of state x.

Certain events in a system may be regarded as *nonprobabilistic* (while still being nondeterministic). The typical example occurs when a user is to choose an input and in the analysis of the system, we do not wish to assign a probability distribution to that choice; in such a case, we regard that choice as nonprobabilistic. All nonprobabilistic choices in the system are lumped into a single choice that is treated as being made by an "adversary" prior to the start of execution. Thus, after this choice is made, the system's execution is purely probabilistic. In Halpern and Tuttle's words, the nonprobabilistic choices have been "factored out".

In the model of computation, each possible choice by the adversary corresponds to a labeled computation tree. In other words, a system is represented as a set of computation trees, each one corresponding to a different choice by the adversary. There is no indication how the adversary's choice is made, just that it is made once and for all, prior to the start of execution.

2.2 Application-Specific System Model

In this section, we impose some additional structure on the general model described in the previous section. We fix the set of agents, fix our model and intuitions regarding commu-

[3]Halpern and Tuttle also include the state of the "environment" as part of the global state. However, we will not be needing this for our application and so we omit it.

nication, place some (environmental) constraints on the agents, and fix the set of choices available to the adversary.

AGENTS As indicated in Figure 1 and the surrounding discussion, we can limit our model to three agents: (1) the system under consideration, denoted Σ, (2) the covert senders (or alternatively, the high environment), denoted $\mathcal{H}$, and (3) the covert receivers (or alternatively, the low environment), denoted $\mathcal{L}$. In the remainder of the paper, we will tacitly assume that the global system is comprised of these three agents.

MODEL OF COMMUNICATION Our model of communication is similar to those of [BC92], [Gra92], and [Mil90]. We view Σ's interface as a collection of channels on which inputs and outputs occur. Since we consider the agent $\mathcal{H}$ (resp., $\mathcal{L}$) to consist of *all* processing that is done in the high (resp., low) environment, including any communication mechanism that delivers messages to Σ, we will not need to model messages in transit or, in Halpern and Tuttle's terminology, the state of the environment; rather, these components of the global state will be included as part of $\mathcal{H}$'s and $\mathcal{L}$'s state.

In many systems of interest, the timing of events is of concern. (See [Lam73] for an early description of covert communication channels that depend on timing; see [Wra92] for more recent work.) In such cases, we model the passage of time by taking the set of times (i.e., the domain of the runs) to be the ticks of some clock that is independent of the covert senders' and receivers' processing. For example, we may think of this clock as being Σ's system clock. In this way, covert channels that depend on time can be properly accounted for.

Since the mechanisms of high-level[4] I/O routines may introduce covert channels (see, e.g., [McC88, §2.3]), we take a very low-level view of I/O. In particular, we assume one input and one output per channel per unit time. That is, for each time we have a vector of inputs (one for each channel) and a vector of outputs (one for each channel). If a given agent produces no new data value at a given time, it may in fact serve as a signal in a covert channel exploitation. Hence, we treat such "no new signal" events as inputs. Similarly, we do not consider the possibility that the system can prevent an input from occurring. Rather, the system merely chooses whether to make use of the input or ignore it. Any acknowledgement that an input has been received is considered to be an output.

Given these considerations, we fix our model of communication as follows. We assume the

[4]In this context, "high-level" means highly *abstract* rather than highly *classified*.

following basic sets of symbols, all nonempty:

C: a finite set of input/output channel names, $c_1, \ldots, c_k$,

I: representing the set of input values,

O: representing the set of output values.

$\mathbb{N}^+$: representing the set of positive natural numbers. This set will be used as our set of "times".

Since there is one input per channel at each time, we will be talking about the vector of inputs that occurs at a given time. We will denote the set of all vectors of inputs by $I[C]$. Typical inputs vectors will be denoted $a, a', a_1, \ldots \in I[C]$.

Similarly, we will denote the set of all output vectors by $O[C]$ and typical output vectors will be denoted $b, b', b_1, \ldots \in O[C]$.

Now, to talk about the history of input vectors up to a given time, we introduce notation for traces. We will denote the set of input traces of length k by $I_{C,k}$. Mathematically, $I_{C,k}$ is a shorthand for the set of functions from $C \times \{1, 2, \ldots k\}$ to I. Therefore, for a trace $\alpha \in I_{C,k}$, we will denote the single input on channel $c \in C$ at time $k' \leq k$ by $\alpha(c, k')$.

We will also need to talk about infinite traces of inputs. For this we use the analogous notation $I_{C,\infty}$, which is short hand for the set of functions from $C \times \mathbb{N}^+$ to I

Similarly, we will denote the set of output traces of length k by $O_{C,k}$ and the set of infinite output traces by $O_{C,\infty}$. Naturally, for an output trace β, $\beta(c, k)$ represents the output on channel c at time k.

There will be situations when we want to talk about vectors or traces of inputs or outputs on some subset of the channels, $S \subseteq C$. In such cases we will use the natural generalizations of the above notations, viz, $I[S]$, $I_{S,k}$, $I_{S,\infty}$, etc..

ENVIRONMENTAL CONSTRAINTS Any given agent will be able to see the inputs and outputs on a subset of the channels. We make this precise by "restricting" vectors and traces to subsets of C. Given an input vector $a \in I[C]$ and a set of channels $S \subseteq C$, we define $a \restriction S \in I[S]$ to be the input vector on channels in S such that $a \restriction S(c) = a(c)$ for all $c \in S$.

Similarly, given an input trace $\alpha \in I_{C,k}$ and a set of channels $S \subseteq C$, we define $\alpha\restriction S \in I_{S,k}$ to be the input trace for channels in S such that $\alpha\restriction S(c,k') = \alpha(c,k')$ for all $c \in S$ and all $k' \leq k$.

We assume that the set of low channels, denoted L, is a subset of C. Intuitively, L is the set of channels that the low environment, $\mathcal{L}$, is able to directly see. In particular, $\mathcal{L}$ is able to see both the inputs and the outputs that occur on channels in L.

In practice, there will be some type of physical or procedural constraints on the agent $\mathcal{L}$ to prevent it from directly viewing the inputs and outputs on channels in $C - L$. For example, those channels may represent wires connected to workstations that are used for processing secret data. In this case, the secret workstations might be located inside a locked and guarded room. In addition, periodic checks of the wires might be made to ensure that there are no wiretaps on them. In this way, $\mathcal{L}$ is prevented from directly viewing the data that passes over the channels in $C - L$.

On the other hand, we place no constraints on the set of channels that $\mathcal{H}$ is able to see. In particular, we make the worst-case assumption that $\mathcal{H}$ is able to see all inputs and outputs on all channels.

The above considerations are consistent with what we've called the "Secure Environment Assumption" in previous work [Gra92, GS92]. In the present paper, this assumption is made precise in terms of our definition of the adversary to be given next.

THE ADVERSARY As discussed above, in Halpern and Tuttle's framework, all nonprobabilistic choices are factored out of the execution of the system by fixing an adversary at the start of execution. To make use of this framework, we must define the set of possible adversaries from which this choice is made.

The "adversary" in our application is the pair of agents, $\mathcal{H}$ and $\mathcal{L}$, that are attempting to send data from the high environment across the system Σ to the low environment. To be fully general, we model these agents as mixed strategies (in the game-theoretic sense). That is, at each point in the execution of the system the strategy gives the probability distribution over the set of next possible inputs, conditioned on the history up to the current point. In the next section, we present an example to motivate the need for such generality. Before doing that, we make the adversary precise with the following two definitions.

Definition 2.1 An *adversary* is a conditional probability function, $\mathcal{A}(a \mid \alpha, \beta, k)$. Here $a \in I[C]$ and k is some time such that there is a time k' with $k \leq k' \leq \infty$, and $\alpha \in I_{C,k'}$ and $\beta \in O_{C,k'}$. (The k indicates that the probability of a is conditional only on the restriction of α and β to k.) Intuitively, the adversary describes the environment's conditional distribution on the next input vector, given the previous history of inputs and outputs. □

Later in this section, we describe how a given adversary $\mathcal{A}$ and the description of a particular system, Σ, are used to construct the corresponding computation tree $T_{\mathcal{A}}$.

Definition 2.2 We say that an adversary $\mathcal{A}$ satisfies the *Secure Environment Assumption with respect to a set of channels* $L \subseteq C$ iff there exists a pair of conditional probability functions $\mathcal{H}$ and $\mathcal{L}$ such that for all $a \in I[C]$, $k \in \mathbb{N}^+$, all $\alpha \in I_{C,k}$, and all $\beta \in O_{C,k}$,

$$\mathcal{A}(a \mid \alpha, \beta, k) = \mathcal{H}(a\restriction(C - L) \mid \alpha, \beta, k) \quad \cdot \quad \mathcal{L}(a\restriction L \mid \alpha\restriction L, \beta\restriction L, k)$$

(where $\cdot$ denotes real multiplication). □

The Secure Environment Assumption can be intuitively understood as saying that the input on channels in $(C - L)$ at time k is (conditionally) statistically independent of the input on channels in L at time k, and the input on channels in L at time k depends only on previous inputs and outputs on channels in L. For the remainder of this paper, we will assume that all adversaries from which the initial choice is made satisfy the Secure Environment Assumption.

Since there is one tree for each possible adversary, we can think of the set of trees as being indexed by the adversaries. Therefore, we will often write $T_{\mathcal{A}}$, $T_{\mathcal{A}'}$, $T_{\mathcal{A}_i}$, etc.

It is clear that for an adversary $\mathcal{A}$ that satisfies the Secure Environment Assumption (wrt L), the conditional probability functions $\mathcal{H}$ and $\mathcal{L}$ that must exist are, in fact, unique. Further, given $\mathcal{H}$ and $\mathcal{L}$, there is a unique adversary, $\mathcal{A}$, for which $\mathcal{H}$ and $\mathcal{L}$ are the probability functions that satisfy the corresponding constraint. We will therefore sometimes write $T_{\mathcal{H},\mathcal{L}}$, $T_{\mathcal{H}',\mathcal{L}'}$, etc. when we want to refer to the parts of the adversary individually.

Note that our definition of an adversary is not meant to be as general as the adversary discussed by Halpern and Tuttle. (In fact, Halpern and Tuttle give no structure at all to their adversary.) Rather, our adversary is application-specific; in particular, it is for

reasoning about multilevel security of probabilistic systems and is not designed to be used outside that domain.

On the other hand, this particular adversary represents a novel application of Halpern and Tuttle's framework. In their examples, the adversary represents one or both of two possible things:

- the initial input to the system; and
- the schedule according to which certain events (e.g., processors taking steps) occur.

In contrast, our adversary does not represent a given input to the system. Rather, it represents a mixed strategy for choosing the inputs to the system. In some sense, we can think of this as a generalization on the first item above; however, our application still fits within the framework set out by Halpern and Tuttle.

THE STATE OF THE SYSTEM At any given point, P, in any given computation tree, $T_{\mathcal{A}}$, there should be a well-defined state of the system. For our purposes, the state includes the following information.

1. All inputs and outputs that have occurred on all channels up to the current time.
2. In [HT93], Halpern and Tuttle make the assumption that all points in all trees are unique. They suggest (and we adopt) the following idea to ensure that this is true. The state encodes the adversary. That is, all nodes in tree $T_{\mathcal{A}}$ encode $\mathcal{A}$. Note that we do *not* assume that any given agent knows the adversary; just that it is somehow encoded in the state. We can think of the high part of the adversary, $\mathcal{H}$, as being encoded in the high environment and the low part, $\mathcal{L}$, as being encoded in the low environment.
3. Typically, there are additional components of the global state representing the internal state of Σ. For example, in describing Σ, it is often convenient to use internal state variables. The state of these variables can be thought of as a vector of values, one value for each state variable. Thus, the internal state, when it exists, will be denoted c, and the history of internal states will be denoted γ.

COMPUTATION TREES Now that we have set out the possible states of the system (i.e., the points of computations), we can talk about the construction of the computation trees.

For each reachable point, P, we assume that Σ's probability distribution on outputs is given. For example, this can be given by a conditional probability distribution, $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$, where c is the vector representing values of all internal state variables (i.e., the internal system state) at time $k+1$, $b \in O[C]$ is the vector of outputs produced by the system at $k+1$, and α, β, γ give the history through k of inputs, outputs, and internal state values, respectively.

Given $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$ and the adversary, $\mathcal{A}$ we can construct the corresponding computation tree by starting with the initial state of the system (i.e., the point at the root of the tree with empty histories of inputs, outputs, etc.) and iteratively extending points as follows.

Let P be a point in the tree with internal system history γ, input history α, and output history β. We will make P' a child of P iff

1. P' is formed from P by modifying the internal system state to c and extending P's input history (output history, resp.) with a (b, resp.); and
2. both $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$ and $\mathcal{A}(a \mid \alpha, \beta, k)$ are positive.

In such cases, we label the arc from P to P' with $\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k) \cdot \mathcal{A}(a \mid \alpha, \beta, k)$, i.e., the system, Σ, and the environment, $\mathcal{A}$, make their choices independently.

RUNS OF THE SYSTEM A run of the system is an infinite sequence of states along a path in one of the computation trees. When we want to talk about the particular run, ρ, and time, k, at which a point P occurs, we will denote the point by the pair (ρ, k). Further, if we wish to talk about the various components of the run, i.e., the trace of the inputs, α, outputs, β, or other variables, γ, we will denote the run by (α, β, γ) and denote the point, P, by $(\alpha, \beta, \gamma, k)$.

For a given tree, T, we denote the set of runs (i.e., infinite sequences of states), formed by tracing a path from the root, by $runs(T)$.

For security applications we are concerned with information flow into and out of the system rather than with information in the system per se. Thus, though our system model is

adequate to represent internal states and traces thereof, in subsequent sections it will be adequate to represent systems entirely in terms of input and output. For example, system behavior at time k can be represented by '$\mathcal{O}(b \mid \alpha, \beta, k)$' rather than '$\mathcal{O}(b, c \mid \alpha, \beta, \gamma, k)$'.

3 Syntax

In this section we set out our formal language and use it to describe two simple systems. Then we give the axioms and rules of our logic.

3.1 Formation Rules

To describe the operation of the system under consideration (viz, Σ), we use a variant of Lamport's Raw Temporal Logic of Actions (RTLA) [Lam91].[5] The primary difference is that we add a modal operator $\Pr_i(\varphi)$ that allows us to specify and reason about the probabilistic behavior of the system.

From the previous section, we assume the following basic sets of symbols, all nonempty: C, I, O, and $\mathbb{R}$. Members of $\mathbb{R}$ will have the usual representation—e.g., $43.5 \in \mathbb{R}$.

We will also be talking about the subjects (or agents) of the system. Formally, a *subject*, $S \subseteq C$, is identified with the process's view of the system, i.e. the set of channels on which it can *see* the inputs and outputs.

Formulae in the language are built up according to the following rules.

- constants from the set of basic symbols are terms.
- state variables (representing the value of that variable in the current state) are terms. Among the state variables, there are two reserved for each communication channel. For each $c \in C$, we have a state variable c_{in} that takes values from I, and another state variable c_{out} that takes values from O. Note that, implicitly, *inputs* are from the covert senders and receivers into the system (Σ) and *outputs* are from the system to the covert senders and receivers. This is because Σ is the system under consideration (i.e., with respect to which we are reasoning about security). We have no mechanism (and

[5] Roughly speaking, Raw Temporal Logic of Actions (RTLA) is the same as Lamport's Temporal Logic of Actions (TLA) without the treatment of stuttering [Lam91]. Since we are not, in this paper, concerned with refinement, we omit the considerations of stuttering and use RTLA.

no need) to specify communication between agents not including the system under consideration.

- primed state variables (e.g., c'_{in}) are terms. (These represent the value of the variable in the next state.)
- We use standard operators among terms (e.g., $+$ and $\cdot$ for addition and multiplication, respectively), with parentheses for grouping subterms, to form composite terms.
- an atomic *predicate* is an equation or inequality among terms *not* containing primed state variables.
- an atomic *action* is an equation or inequality among terms (possibly including primed as well as unprimed state variables). (Note that all predicates are actions.)
- for any action, φ, and for any subject $S \subseteq C$, $Pr_S(\varphi)$ is a real-valued term (representing the subjective *probability* that S assigns to the formula φ).
- For any predicate, φ, φ is a temporal formula.
- For any action or temporal formula φ, $\Box\varphi$ is a temporal formula (to be read intuitively as *always* φ).
- We build up composite predicates, actions, and temporal formulae, resp., in the usual recursive fashion using $\wedge$, $\vee$, $\neg$, and $\rightarrow$.

Now, to specify and reason about our security properties of interest, we add three finite sets of modal operators on formulae: $k_1, \ldots, k_n$, $\mathbf{K}_1, \ldots, \mathbf{K}_n$, and $R_1, \ldots, R_n$, representing knowledge of a (relatively) weak adversary, knowledge of a powerful adversary, and permitted-knowledge respectively for each subject (represented by the subscript of the operator). Therefore, we add the following additional formation rules to our syntax.

- For any action (temporal formula, resp.) φ, and for any subject $S \subseteq C$, $k_S(\varphi)$ (representing that the weak adversary S knows φ), $\mathbf{K}_S(\varphi)$ (representing that the powerful adversary S knows φ) and $R_S(\varphi)$ (representing that S has *permitted* knowledge of φ) are actions (temporal formulae, resp.).

Later in the paper, we will make the meaning of these three operators precise. For now, we merely mention that the weak-adversary knowledge operators (k_S) will be given the standard semantics (e.g., as in [HT93]); the powerful-adversary knowledge operators ($\mathbf{K}_S$) will be given semantics that imply greater knowledge on the part of the subject (viz, knowledge of the probability of certain future events).

3.2 Examples

We now give two simple examples of how to describe systems in our language. Ultimately, we will have sufficient formal machinery to show that one of these systems is secure and the other is not; however, here we simply set them out formally. These descriptions are meant to give the reader an intuitive feel for the meaning of expressions in the language. Precise meanings will be given in §4. Also, the second of these examples will motivate our choice of modeling adversaries as strategies.

Example 3.1 The first example is a simple encryption box that uses a "one-time pad" [Den82]. It has two channels, *high* and *low*. At each tick of the system clock, it inputs a 0 or 1 on the high channel and outputs a 0 or 1 on the low channel. The low output is computed by taking the "exclusive or" (denoted $\oplus$) of the high input and a randomly generated bit. It is well known that this results in an output stream that is uniformly distributed. Therefore, we can describe this system as follows.

Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. Then, the system is specified by the following formula.

$$\Box\,(Pr(l'_{out} = 0) = Pr(l'_{out} = 1) = 0.5)$$

In this formula, l_{out} is a state variable representing the output on the low channel, l. Therefore, l'_{out} is the output on l at the *next* time. Further, $Pr(l'_{out} = 0)$ denotes the probability that the output on l is a 0 at the next time. Hence, the entire formula says that at all times, the probability of Σ producing a one (1) on the next clock tick is equal to the probability of producing a zero (0), which is equal to 0.5. Note that we have not specified inputs per se since these constitute environment behavior rather than system behavior.

□

Example 3.2 The second example is an insecure version of the simple encryption box. This system was first described by Shannon in [Sha58].

As in the first example, at each tick Σ computes the "exclusive or" of the current high input and a randomly generated bit and outputs that value on the low channel. However, in this system, the randomly generated bit used at any given tick is actually generated and output on the high output channel during the *previous* tick of the clock.

This can be expressed in our formalism as follows. Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. The following formula specifies the system.

$$\Box(Pr(h'_{out} = 0) = Pr(h'_{out} = 1) = 0.5 \wedge l'_{out} = h_{out} \oplus h'_{in})$$

Note that in the second conjunct, h_{out} is unprimed, indicating that the output on l at the *next* time is the "exclusive or" of the *current* output on h with the *next* input on h.

Now note that if the high agent ignores its output, then this system acts exactly as the system from the previous example (and can be used for perfect encryption). In particular, suppose we were to model an adversary as an input string—the input to be provided by the high agent. Then, it is easy to prove that for any adversary (i.e., any high input string) fixed prior to the start of execution, the output to low will be uniformly distributed and, in fact, will contain no information about the high input string.

However, the bit that will be used as the one-time pad at time t is available to the high agent at time $t-1$. Therefore, (due to the algebraic properties of "exclusive or", viz, $x \oplus x \oplus y = y$) the high agent can use this information to counteract the encryption. In particular, the high agent can employ a (game-theoretic) strategy to send any information it desires across the system to the low agent.

For example, suppose the high agent wishes to send a sequence of bits, $b_1, b_2, \ldots$. We'll denote the high input (resp., output) at time k by $h_{in}(k)$ (resp., $h_{out}(k)$). The appropriate strategy for the high agent is as follows.

The high agent chooses its input for time $k+1$ as $h_{in}(k+1) = h_{out}(k) \oplus b_k$.

Thus, the output to low at time $k+1$, denoted $l_{out}(k+1)$ is computed as follows.

$l_{out}(k+1) = h_{out}(k) \oplus h_{in}(k+1)$ [by the system description]

$$= h_{out}(k) \oplus h_{out}(k) \oplus b_k \qquad \text{[by the high strategy]}$$
$$= b_k \qquad \text{[by the properties of } \oplus\text{]}$$

Thus, by employing the correct strategy, the high agent can noiselessly transmit an arbitrary message over Σ to the low agent. This, of course, motivates our choice of strategies as the adversary, rather than, e.g., input strings.

□

We now have some sense of the formal language, with the exception of the modal operators k_S, $\mathbf{K}_S$, and R_S. As previously mentioned, these operators will be used to formalize the security property that interests us; so, we will illustrate their use in a later section. First, we will describe the logical axioms and inference rules that are used to prove properties about systems.

3.3 The Logic

We now give the axioms of our logic. In the following, we will use 'φ' and 'ψ' to refer to formulae of our language.

Propositional Reasoning All instances of tautologies of propostional logic.

Temporal Reasoning The following are standard axioms for temporal reasoning about discrete systems. The logic they constitute is generally called S4.3*Dum* or sometimes **D**. (See [Gol92] for details. Note also that these are the formulae Abadi uses to axiomatize Lamport's TLA [Aba90].) We have labelled the axioms with their historical names. Let φ and ψ be formulae of our language.

K $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$

4 $\Box\varphi \rightarrow \Box\Box\varphi$

D $\Box\varphi \rightarrow \Diamond\varphi$

L $\Box(\varphi \wedge \Box\varphi \rightarrow \psi) \vee \Box(\psi \wedge \Box\psi \rightarrow \varphi)$

Z $\Box(\Box\varphi \rightarrow \varphi) \rightarrow (\Box\Diamond\varphi \rightarrow \Box\varphi)$

'$\Diamond\varphi$' can be interpreted roughly as saying that at some point φ is true. Formally, it is viewed as notational shorthand: for all formulae φ, $\Diamond\varphi \triangleq \neg\Box\neg\varphi$. **K** basically guarantees that the temporal operator respects modus ponens. Each of the other axioms captures a feature of time that we desire. **4** gets us transitivity. **D** guarantees that we don't run out of time points (seriality). **L** guarantees that all points in time are connected. And, **Z** guarantees that time is discrete. (Between any two points in time there are at most finitely many other points.)

Real Number Axioms Standard field and order axioms for the real numbers (to apply to members of $\mathbb{R}$ and function terms with range $\mathbb{R}$.) We will not enumerate these axioms. (See any elementary real analysis book for enumeration, e.g., [Mar74] or [Rud].)

Epistemic Reasoning The (nonredundant) axioms of the Lewis system **S5**. (cf. [Che80] or [Gol92]) apply to the strong knowledge operators ($\mathbf{K}_i$), the weak knowledge operators (k_i), and the permitted-knowledge operators (R_i). We state them only for the (strong) knowledge operators. As for temporal axioms, we give the axioms their historical names. Let S be a subject, and let φ and ψ be formulae of our language.

K $[\mathbf{K}_S(\varphi) \wedge \mathbf{K}_S(\varphi \rightarrow \psi)] \rightarrow \mathbf{K}_S(\psi)$ (Knowledge respects modus ponens.)

T $\mathbf{K}_S(\varphi) \rightarrow \varphi$ (What one knows is true.)

5 $\neg\mathbf{K}_S(\varphi) \rightarrow \mathbf{K}_S\neg\mathbf{K}_S(\varphi)$ (If you don't know something, then you know that you don't know it.)

We also have two axioms for relating weak knowledge to permitted knowledge and permitted knowledge to strong knowledge.

kR $k_S(\varphi) \rightarrow R_S(\varphi)$

RK $R_S(\varphi) \rightarrow \mathbf{K}_S(\varphi)$

Random Variable Axioms The standard requirements for random variables (in the probability theoretic sense).

PM (Positive Measure) for any formula, φ, and any subject, S, $Pr_S(\varphi) \geq 0$ (The probability of any event is greater than or equal to zero.)

NM (Normalized Measure) for any channel, c, and any subject, S,

$\sum_{a \in I} Pr_S(c_{in} = i) = 1$ (The probability of all possibilities sums to one.)

$\sum_{b \in O} Pr_S(c_{out} = o) = 1$

Input/Output Axioms for knowledge and permitted-knowledge of inputs and outputs. Let S be a subject, let $c \in S$ be a channel that is visible to S, and let $a \in I$ be an input, $b \in O$ be an output, and $r \in \mathbb{R}$ be a real number.

KO $Pr_S(c'_{out} = o) = r \rightarrow \mathbf{K}_S(Pr_S(c'_{out} = o) = r)$

RI $Pr_S(c'_{in} = i) = r \rightarrow R_S(Pr_S(c'_{in} = i) = r)$

Intuitively, **KO** say that a subject knows the distribution on its own outputs conditioned on the previous history of inputs and outputs that it has seen. Similarly, a subject knows the distribution on its own inputs conditioned on the previous history of inputs and outputs it has seen. However, we need no corresponding axiom **KI** since it follows trivially from **RI** and **RK**. From theorems **KI** and **KO** we can inductively show that every subject knows the probability of any event that it can see in finite time. **RI** says that a subject is permitted to know the conditional distribution on its own inputs. But, a subject is permitted to know the conditional distribution on its own outputs only if the system is secure—e.g., for a low subject, only if knowing that distribution does not reveal any information about the distribution on high inputs. The absence of an axiom **RO**, corresponding to **KO**, is what syntactically captures this.

The above are all of our axioms. We now give the rules of our logic, which are all standard.

MP (Modus Ponens) From φ and $\varphi \rightarrow \psi$ infer ψ.

Nec (Necessitation) This rule applies to all of the modal operators we have introduced: $\Box$, $\mathbf{K}_S$, k_S, and R_S. (It is called 'necessitation' because it was originally applied to a necessity operator.) We set it out for $\Box$ only. From $\vdash \varphi$ infer $\vdash \Box\varphi$

Note that in the above, '$\vdash \varphi$' indicates a derivation of φ from the axioms alone, rather than from a set of premises. (Derivations will be formally defined below.) Thus, in the case of knowledge (strong or weak) for example, **Nec** says that if φ is a theorem (derivable without any premises) then all subjects know φ.

We now have sufficient machinery to give a characterization of a formal derivation.

Definition 3.3 Let Γ be a finite set of formulae of our language. A finite sequence of formulae $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_n$ is called a *derivation* (of φ_n from Γ) iff each φ_k $(k = 1, \ldots, n)$ satisfies one of the following:

- $\varphi_k \in \Gamma$
- φ_k is an axiom.
- φ_k follows from some theorem by **Nec**.
- For some $i, j < k$, φ_k results from φ_i and φ_j by **MP**.

We write '$\Gamma \vdash \varphi$' to indicate a derivation of φ from Γ, and we write '$\vdash \varphi$' to indicate a derivation of φ from the axioms alone. □

This completes our statement of the formal system.

4 Semantics

In the last section we presented a syntactic system. So far we have only intuitive meanings to attach to this formalism. In this section we provide semantics for our system in terms of the Halpern-Tuttle framework and our application-specific model set out in §2.

4.1 Semantic Model

A *model* M is a tuple of the form:

$$\langle \mathbb{R}, +, \cdot, \leq, W, \mathcal{T}, C, I, O, v, \kappa_1^{powerful}, \ldots, \kappa_{|\mathcal{P}(C)|}^{powerful}, \kappa_1^{weak}, \ldots, \kappa_{|\mathcal{P}(C)|}^{weak}, \delta_1, \ldots, \delta_{|\mathcal{P}(C)|}, \rangle$$

Here, $\mathbb{R}$ and its operations and ordering relation gives us the real numbers; W is the set of worlds (i.e., global states); $\mathcal{T}$ is the set of labeled computation trees (with nodes from W); C, I, and O are the sets of channels, possible inputs, and possible outputs, respectively; v is the assignment function, which assigns semantic values to syntactic expressions at each world; (values of v at a particular world P, will be indicated by the projection 'v_P'); the $\kappa_{i_S}^{powerful}$ and $\kappa_{i_S}^{weak}$ are knowledge accessibility relations, one each for each subject S; and the δ_{i_S} are permitted-knowledge accessibility relations, also one for each subject. In the

remainder of this paper we will generally denote the accessibility relations corresponding to subject S by '$\kappa_S^{powerful}$', 'κ_S^{weak}', and 'δ_S'. These will each be further explained when we come to the assignment function.

In assigning meaning to our language, it is of fundamental importance to associate a probability space with each labeled computation tree. In particular, for each labeled computation tree $T_{\mathcal{A}}$ we will construct a sample space of runs, $\mathcal{R}_{\mathcal{A}}$, an event space, $\mathcal{X}_{\mathcal{A}}$ (i.e., those subsets of $\mathcal{R}_{\mathcal{A}}$ to which a probability can be assigned) and a probability measure $\mu_{\mathcal{A}}$ that assigns probabilities to members of $\mathcal{X}_{\mathcal{A}}$.

Our construction of this probability space is quite natural and standard (see, e.g., [Sei92] as well as [HT93] for two instances). We will not go into detail explaining the basic concepts of probability and measure theory here (cf. [Hal50] or [Shi84]).

Definition 4.1 For a labeled computation tree $T_{\mathcal{A}}$, the associated **sample space** $\mathcal{R}_{\mathcal{A}}$ is the set of all infinite paths starting from the root of $T_{\mathcal{A}}$.

The set $e \subseteq \mathcal{R}_{\mathcal{A}}$, is called a **generator** iff it consists of the set of all traces with some common finite prefix. The generators are the probability-theoretic events corresponding to finite traces. We can now define the **event space**, $\mathcal{X}_{\mathcal{A}}$, to be the (unique) field of sets generated by the set of all generators (i.e., $\mathcal{X}_{\mathcal{A}}$ is the smallest subset of $\mathcal{P}(\mathcal{R}_{\mathcal{A}})$ that contains all of the generators and is closed under countable union and complementation).

Suppose e is a generator corresponding to the finite prefix given by (ρ, k). Then, the probability measure, $\mu_{\mathcal{A}}$, is defined for e as the product of the transition probabilities from the root of the tree, along the path ρ, up to time k. Further, there is a unique extension of $\mu_{\mathcal{A}}$ to the entire event space [Hal50]. □

4.2 Assignment Function

For a given point, P, we will assign truth values to temporal formulae φ at this point. In addition, we assign values to variables, for example the input on a channel, at this point. The assignment function that does both of these is denoted by v_P.

To define v_P, we will need to assign truth values to action and temporal formulae. Therefore we will also define functions $v_{(P_1,P_2)}$ (where P_1 and P_2 are points) and v_ρ (where ρ is a run) to assign truth values to action formulae over a pair of points and temporal formulae on a run, respectively.

We define v_P, $v_{(P_1,P_2)}$, and v_ρ mutually recursively below. First we present some additional notation.

Since nodes are unique even across trees, for a given node P, there is no ambiguity in referring to "the tree that contains P". In the following, we will use $tree(P)$ to denote that tree.

We use the notation $succ(P)$ to denote the set of nodes that succeed P in $tree(P)$.

We use the notation $extensions(P)$ to denote the set of infinite sequences of states starting at P in $tree(P)$.

As discussed in [HT93], to each subject, S, and point, P, we need to associate a sample space, $\mathcal{S}_{S,P}$. Each such sample space will be a set of points from $tree(P)$. Intuitively, these are the points (within the tree that contains the current execution) that the subject S considers possible. We will set out these sample spaces below. For the time being, we simply make use of the notation $\mathcal{S}_{S,P}$ to refer to them.

We will be rather abusive in the use of our probability measures $\mu_{\mathcal{A}}$. In particular, when we have a finite set of points, x, we will write $\mu_{\mathcal{A}}(x)$ to denote the probability (as assigned by $\mu_{\mathcal{A}}$) of passing through one of the points in x. Technically, this is wrong, since $\mu_{\mathcal{A}}$ is defined for (certain) sets of runs; not for sets of points. However, the mapping between the two is extremely natural; the set of runs correspondings to a point is the set of runs that *pass through* that point. Further, by the construction of our probability spaces, all sets of runs corresponding to finite sets of points are measureable. Therefore, there is no danger in this abuse of notation and it greatly simplifies our presentation.

As is standard (see, e.g., [HT93]), we will be using *accessibility relations*—one for each subject—on points to give semantics to our three knowledge operators. We define these relations below. For the time being, we simply make use of the notation $\kappa_S^{powerful}$ to refer to the powerful-adversary knowledge accessibility relation, κ_S^{weak} to refer to the weak-adversary knowledge accessibility relation, and δ_S to refer to the permitted-knowledge accessibility relation.

We now define v_P, $v_{(P_1,P_2)}$, and v_ρ. Let P be a point at time k in the execution $\rho = (\alpha, \beta, \gamma)$ in computation tree $T_{\mathcal{A}}$.

- Numbers are assigned to number names.
- Members of I and O are assigned to their syntactic identifiers.

- For any channel $c \in C$,

$$v_P(c_{in}) \triangleq \alpha(c,k)$$

- For any channel $c \in C$,

$$v_P(c_{out}) \triangleq \beta(c,k)$$

- For any variable name, X, excluding channel variables (such as c_{in} or c_{out})

$$v_P(X) \triangleq \gamma(X,k)$$

- To assign truth values to actions, we need to assign values to terms at *pairs of points*. Constants do not change their values when we move to pairs of points. However, primed and unprimed variables are evaluated differently. For any state variable, X, and any pair of points (P_1, P_2),

$$v_{(P_1,P_2)}(X) \triangleq v_{P_1}(X)$$

In contrast,

$$v_{(P_1,P_2)}(X') \triangleq v_{P_2}(X)$$

$$v_{(P_1,P_2)}(\varphi) \triangleq v_{P_1}|_{P_2}(\varphi)$$

where $v_{P_1}|_{P_2}(\varphi)$ follows v_{P_1} except that all primed terms are assigned according to v_{P_2}.

- Composite terms are assigned values at a point and at a pair of points in the natural way. For example,

$$v_P(X+Y) \triangleq v_P(X) + v_P(Y)$$

and

$$v_{(P_1,P_2)}(X+Y) \triangleq v_{(P_1,P_2)}(X) + v_{(P_1,P_2)}(Y)$$

- Similary, predicates and action formulae are assigned truth values at a point and at a pair of points, respectively, in the natural way. For example,

$$v_P(X \leq Y) = \textbf{true} \quad \text{iff} \quad v_P(X) \leq v_P(Y)$$

and

$$v_{(P_1,P_2)}(\varphi \wedge \psi) = \textbf{true} \quad \text{iff} \quad v_{(P_1,P_2)}(\varphi) = \textbf{true} \text{ and } v_{(P_1,P_2)}(\psi) = \textbf{true}$$

- An action formula, φ, is true at a point, P, iff it is true for all pairs of points emanating from P. More precisely,

$$v_P(\varphi) = \textbf{true} \quad \text{iff} \quad \forall P' \in succ(P),\ v_{(P,P')}(\varphi) = \textbf{true}$$

(Since we have not needed to include quantification in our language we are free to use '$\forall$' and '$\exists$' as metalinguistic shorthand.)

- To interpret the *probability* of an action φ at a point P, we will take the set of all pairs of points, (P_1, P_2) emanating from points in $\mathcal{S}_{S,P}$. Restricting to this set, we compute the probability of those pairs such that $v_{(P_1,P_2)}(\varphi)$ evaluates to true. More precisely, for any action formula, φ, and for any subject $S \subseteq C$,

$$v_P(Pr_S(\varphi)) \triangleq \mu_{\mathcal{A}}(P)(\mathcal{S}_{S,P}(\varphi))$$

where

$$\mathcal{S}_{S,P}(\varphi) \triangleq \{P_2 \mid \exists P_1 \in \mathcal{S}_{S,P} \wedge P_2 \in succ(P_1) \wedge v_{(P_1,P_2)}(\varphi) = \textbf{true}\ \}$$

and $\mathcal{A}(P)$ is the adversary corresponding to $tree(P)$.

- For any predicate, φ, and run, ρ,

$$v_\rho(\varphi) \triangleq v_{\rho(1)}(\varphi)$$

- For any (action or temporal) formula, φ, and run, ρ,

$$v_\rho(\Box\varphi) = \textbf{true} \quad \text{iff} \quad \forall i, v_{\rho(i)}(\varphi) = \textbf{true}$$

- A temporal formula is true at a point iff it is true in all runs extending from that point. More precisely, for any temporal formula, φ,

$$v_P(\varphi) \triangleq \forall \rho \in extensions(P), v_\rho(\varphi)$$

- Composite action formulae and temporal formulae are assigned truth values at points in the natural way. For example,

$$v_P(\varphi \wedge \psi) = \textbf{true} \quad \text{iff} \quad v_P(\varphi) = \textbf{true} \ \text{ and } v_P(\psi) = \textbf{true}$$

- Our three knowledge operators are all S5 modal operators and are given semantics in terms of the accessibility relations (on points) in the standard way; viz, for powerful-adversary knowledge,

$$v_P(\mathbf{K}_S(\varphi)) = \textbf{true} \quad \text{iff} \quad \forall P',\ \kappa_S^{powerful}(P,P') \Rightarrow v_{P'}(\varphi) = \textbf{true}$$

for weak-adversary knowledge,

$$v_P(k_S(\varphi)) = \textbf{true} \quad \text{iff} \quad \forall P',\ \kappa_S^{weak}(P,P') \Rightarrow v_{P'}(\varphi) = \textbf{true}$$

and for permitted knowledge,

$$v_P(R_S(\varphi)) = \textbf{true} \quad \text{iff} \quad \forall P',\ \delta_S(P,P') \Rightarrow v_{P'}(\varphi) = \textbf{true}$$

To complete our semantics for probability formulas, we need to choose the sample spaces $\mathcal{S}_{S,P}$ for each subject at each point. Our approach is quite straightforward. We will choose $\mathcal{S}_{S,P}$ to be the set of points within $tree(P)$ that have the same history of inputs and outputs on channels S as occur on the path to point P. More precisely, we have the following definitions.

Definition 4.2 Let $S \in C$ be a subject and let $\rho_1 = (\alpha_1, \beta_1, \gamma_1)$ and $\rho_2 = (\alpha_2, \beta_2, \gamma_2)$ be two runs (not necessarily in the same tree). We say that ρ_1 and ρ_2 *have the same S-history up to time k* if and only if

$$\forall i, 1 \leq i \leq k,\ \forall c \in S, \quad \alpha'(c,i) = \alpha(c,i) \ \wedge \ \beta'(c,i) = \beta(c,i)$$

□

Definition 4.3 Let $S \in C$ be a subject and let $P_1 = (\rho_1, k_1)$ and $P_2 = (\rho_2, k_2)$ be two points (not necessarily in the same tree). We say that P_1 and P_2 *have the same S-history* if and only if the following two conditions hold.

1. $k_1 = k_2$.
2. ρ_1 and ρ_2 have the same S-history up to time k_1.

□

Definition 4.4 Let $S \in C$ be a subject and P be a point; the *sample space for S at point P* is given by

$$\mathcal{S}_{S,P} \triangleq \{\ P' \mid tree(P') = tree(P) \ \wedge \ P' \text{ and } P \text{ have the same } S\text{-history}\ \}$$

□

In a more general setting, we would also want to consider the possibility that a subject S has internal state variables and could use these to make finer distinctions between points. However, in our application, all of the internal processing of the relevant subjects (viz, $\mathcal{H}$ and $\mathcal{L}$) is encoded in the adversary and is thus factored out of the computation tree. We therefore do not lose any needed generality in making the above definition.

Now, to complete our description of the assignment function we need only describe the relations $\kappa_S^{powerful}$, κ_S^{weak}, and δ_S for all $S \subseteq C$.

Definition 4.5 Our definition of κ_S^{weak} (and hence our definition of weak-adversary knowledge) is the standard definition of knowledge in a distributed system. In particular, for any two points, P_1 and P_2 (not necessarily in distinct trees) and any subject, $S \subseteq C$, We say that P_2 is *weak-adversary-accessible* from P_1, denoted '$\kappa_S^{weak}(P_1, P_2)$' if and only if P_1 and P_2 have the same S-history. □

Our definition of $\kappa_S^{powerful}$ (and hence, our definition of powerful-adversary knowledge) is novel. In the analysis of distributed protocols and in other areas of computer science, it is typical to use the above weak-adversary knowledge accesibility relation (or something roughly equivalent). Our definition of accessibility for *powerful-adversary* knowledge will require more—in other words, using this definition subjects *know* more. In particular, subjects "know" the probability distribution over the future inputs and outputs on the channels that they can see. That is, if the probability of a given future output on a low channel is x, then (assuming a powerful adversary) the low environment knows that. To make this notion precise, we need some definitions.

Definition 4.6 Let $S \subseteq C$ be a subject and let e be a set of runs, $\{\rho_i\}$, (not necessarily taken from any one computation tree). We say that e is an *S-event* if and only if there exists a time $k \in \mathbb{N}^+$ such that for any two runs, ρ_1 and ρ_2, having the same S-history up to time k, $\rho_1 \in e$ iff $\rho_2 \in e$.

For an S-event, e, we will refer to the least k such that above condition holds as the *length of* e. □

Intuitively, an event e is an S-event if and only if there is some finite time k (i.e., its length) after which S can always determine whether or not e has occurred.

Note that in general, an S-event contains runs from more than one computation tree. Therefore, such "events" will not be measurable in any of our probability spaces. Rather, we think of them as *meta events* and we will be interested in the measure of the subset of the runs that are contained in a given computation tree. To make this precise, we introduce the following definition.

Definition 4.7 Given a computation tree, $T_{\mathcal{A}}$, and an S-event, e, the *projection of* e *onto* $T_{\mathcal{A}}$, denoted $e_{\mathcal{A}}$, is given by:

$$e_{\mathcal{A}} \triangleq runs(T_{\mathcal{A}}) \cap e$$

□

Observation 4.8 Every projection of every S-event is measurable. That is, for any S-event, e, and any computation tree, $T_{\mathcal{A}}$,

$$e_{\mathcal{A}} \in \mathcal{X}_{\mathcal{A}}$$

This is due to the restriction on S-events that they be observable within some finite time. In particular, the projection of an S-event onto a tree, T, must also be observable within a finite time and so, it must be formable from a finite number of unions and complementations of the generators of T. □

Now we are ready to give the definition of the knowledge accessibility relation.

Definition 4.9 Let P_1 and P_2 be two points in (not necessarily distinct) trees $T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$, respectively and let $S \subseteq C$ be a subject. We say that P_2 is *powerful-adversary-accessible* from P_1, denoted '$\kappa_S^{powerful}(P_1, P_2)$' if and only if

1. P_1 and P_2 have the same S-history; and
2. for any S-event e, $\mu_{\mathcal{A}_1}(e|\mathcal{S}_{S,P_1}) = \mu_{\mathcal{A}_2}(e|\mathcal{S}_{S,P_2})$

□

Thus, when two points are $\kappa_S^{powerful}$-accessible, this implies not only that the two points have the same S-history, but also, conditioned on the current S-history, the probability distribution on all S-events, including future events, is the same. As mentioned previously, using this definition, subjects "know more" than when using the standard definition. However, we view this as another case where we've adopted the worst-case scenario; that is, we've given the penetrators, $\mathcal{H}$ and $\mathcal{L}$, the greatest conceivable knowledge at any given point in the execution of the system. We will see later in the paper that this choice corresponds to some existing information-theoretic definitions of perfect multilevel security.

Our definition of permitted knowledge is also novel. From our viewpoint, a subject's permitted knowledge does not change over the course of the system's execution. That is, a given subject's permitted knowledge is set *prior* to the start of execution. (It is only a subject's *knowledge* that changes during the system's execution.) Thus, we can capture a subject's permitted knowledge by defining an accessibility relation on computation trees. We will say that two *points* are accessible if and only if they have the same S-history and their two containing trees are accessible; roughly speaking, two computation trees, $T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$, will be accessible if and only if the parts of the adversaries, $\mathcal{A}_1$ and $\mathcal{A}_2$, that correspond to S "act the same" in both trees. We make this precise as follows.

Definition 4.10 Let S be a subject and $T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$ be two computation trees. We say that $T_{\mathcal{A}_2}$ is Δ_S-*accessible* from $T_{\mathcal{A}_1}$, denoted '$\Delta_S(T_{\mathcal{A}_1}, T_{\mathcal{A}_2})$' if and only if for any point P_1 in $T_{\mathcal{A}_1}$ there is a point P_2 in $T_{\mathcal{A}_2}$ such that

1. P_1 and P_2 have the same S-history; and
2. for any channel $c \in S$ and any input $i \in I$, $v_{P_1}(Pr_S(c'_{in} = i)) = v_{P_2}(Pr_S(c'_{in} = i))$.

□

Definition 4.11 Let S be a subject and P_1 and P_2 be two points. We say that P_2 is δ_S-*accessible* from P_1, denoted '$\delta_S(P_1, P_2)$' if and only if

1. P_1 and P_2 have the same S-history; and

2. $\Delta_S(tree(P_1), tree(P_2))$.

□

Thus, the δ_S relation reflects the fact that subjects are permitted to know the conditional probability distribution on their inputs: two points are δ_S-accessible (i.e., as far as S is *permitted to know* they are the same point) if and only if the conditional distribution on inputs visible to S is the same at both points.

There is a close relationship between our definition of permitted knowledge and the Secure Environment Assumption. In particular, recall that for any adversary, $\mathcal{A}$, that satisfies the Secure Environment Assumption wrt L (viz, definition 2.2), there is a one-to-one correspondence between $\mathcal{A}$ and the two components of the environment, $\mathcal{H}$ and $\mathcal{L}$.

Let $\mathcal{A}_1 = (\mathcal{H}_1, \mathcal{L}_1)$ and $\mathcal{A}_2 = (\mathcal{H}_2, \mathcal{L}_2)$ be two adversaries that satisfy the Secure Environment Assumption wrt L. Since the low environment determines the probabilities with which inputs occur on channels in L, it is clear that $\Delta_S(T_{\mathcal{A}_1}, T_{\mathcal{A}_2})$ if and only if $\mathcal{L}_1 = \mathcal{L}_2$.

Intuitively, this relationship can be understood as follows. A subset, L, of the interface of Σ has been partitioned off. By our definition of permitted knowledge, we will say that the low environment, $\mathcal{L}$, is permitted to know how the inputs on L are chosen, but *not* how other (high) inputs are chosen. By the Secure Environment Assumption, we are saying that $\mathcal{L}$ cannot get any information about how high inputs are chosen via any means outside of Σ. With these two definitions in place, we have effectively isolated the question that interests us, "Can the low environment ($\mathcal{L}$) come to know, via the system of interest (Σ), something about the activity of the high environment ($\mathcal{H}$)?"

In the remainder of the paper, for a point P, formula φ, and set of formulae Γ, we will use '$P \models \varphi$' to indicate that φ is true at P, and $P \models \Gamma$ to indicate that all members of Γ are true at P. Finally, we will use '$\Gamma \models \varphi$' to indicate that φ is true at all worlds at which all members of Γ are true.

5 Soundness

In §6 and §7 below we give a syntactic characterization of security and show that the semantic interpretation of our syntactic characterization of security is equivalent to certain previously

developed characterizations. However, the significance of these results is greatly reduced unless the logic is sound. For, without soundness there is no guarantee that any formal proof of security we might give for a system implies any independently motivated notion of security. A soundness theorem gives us just such a correspondence.

Theorem 5.1 [Soundness] Given a set of formulae of our language Γ and a formula φ,

$$\text{If } \Gamma \vdash \varphi, \text{ then } \Gamma \models \varphi.$$

Proof: In order to prove soundness we must show that the axioms are valid and the rules are truth preserving (except **Nec** which need only be theorem preserving). For most of the axioms and all of the rules the results are completely standard. (Cf. [Che80] and [Gol92].) Hence, we do not set them out here. We specifically assumed a semantics in which all the rules and axioms concerning logical connectives preserve soundness. Since we assume the real numbers are part of our models, the axioms concerning them must all be valid. Likewise, because the $Pr(\varphi)$ terms are interpreted as conditional probabilities of events, the **RV** axioms are valid in our semantics since they reflect basic facts about probability measures. The accessibility relations, set out above in §4, are clearly equivalence relations. Thus, by a standard result of modal logic, the **S5** axioms are all valid and **Nec** (for the knowledge operators) is theorem preserving (cf. [Che80]). The temporal reasoning axioms are similarly valid and **Nec** for the temporal operator is theorem preserving based on the time structure of our model of computation (cf. [Gol92]). Validity of **kR** is immediate and that of **RK** is direct from the definition of an S-event. Therefore, the only axioms that need be checked are the I/O axioms. Let $S \subseteq C$ be a subject, $c \in S$ a channel, $i \in I$ an input, $o \in O$ an output, and $r \in \mathbb{R}$ be a real number.

RI $Pr_S(c'_{in} = i) = r \rightarrow R_S(Pr_S(c'_{in} = i) = r)$

Given a world P_1, suppose that $v_{P_1}(Pr_S(c'_{in} = i)) = r$. Let P_2 be a world such that $\delta_S(P_1, P_2)$. Then P_1 and P_2 have the same S-history and $\Delta_S(tree(P_1), tree(P_2))$. Thus, there exists a point $P'_2 \in tree(P_2)$ such that P_1 and P'_2 have the same S-history, and $v_{P'_2}(Pr_S(c'_{in} = i)) = v_{P_1}(Pr_S(c'_{in} = i)) = r$. But, the definition of $v_P(Pr_S(\varphi))$ guarantees that if there is such P'_2 then for any $P \in tree(P_2)$ that has the same S-history as P_1, $v_P(Pr_S(c'_{in} = i)) = r$, in particular $v_{P_2}(Pr_S(c'_{in} = i)) = r$. So, by the truth conditions for R_S, $v_{P_1}(R_S(Pr_S(c'_{in} = i)) = r) =$ **true** . So, by truth conditions for the conditional, **RI** is true at every world P, hence valid.

KO $Pr_S(c'_{out} = o) = r \rightarrow \mathbf{K}_S(Pr_S(c'_{out} = o) = r)$

Given a world P_1, suppose that $v_{P_1}(Pr_S(c'_{out} = o)) = r$. Let P_2 be a world such that $\kappa_S^{powerful}(P_1, P_2)$. Since $c \in S$, $c'_{out} = o$ is clearly an S-event. So, by definition of the $\kappa_S^{powerful}$ relation, $v_{P_2}(Pr_S(c'_{out} = o)) \triangleq \mu_{\mathcal{A}_2}(\mathcal{S}_{S,P_2}(c'_{out} = o) = \mu_{\mathcal{A}_1}(\mathcal{S}_{S,P_1}(c'_{out} = o) \triangleq v_{P_1}(Pr_S(c'_{out} = o)) = r$. So, by the truth conditions for $\mathbf{K}_S$, $v_{P_1}(\mathbf{K}_S(Pr_S(c'_{out} = o)) = r) =$ **true** . So, by truth conditions for the conditional, **KO** is true at every world P, hence valid.

□

This completes our discussion of the logic itself. In the remainder of the paper we focus on security and applications of the logic thereto.

6 Formal Definition of Security

In this section, we give a definition of security—which we call the *Syntactic Security Condition* (SSC)—using the powerful-adversary-knowledge and permitted-knowledge operators of our logic. This definition is based on the definition of "Causality" given by Bieber and Cuppens [BC92], which was based on the work of Glasgow, MacEwen, and Panangaden [GMP90]. Although the statement of SSC is almost syntactically identical to Bieber and Cuppens' definition of Causality, due to the differences in the semantics of the respective logics, the meanings of (i.e., the semantic interpretations of) SSC and Causality are different. In fact, it is straightforward to show that for deterministic systems, the meaning of SSC is equivalent to the meaning of Causality. Thus, since SSC additionally applies to probabilistic systems, SSC can be viewed as a generalization of Causality. In the second subsection, we show that the meaning of SSC is equivalent to the definition of Probabilistic Noninterference given in [Gra92].

6.1 The Syntactic Security Condition

For a given subject L, the syntactic security condition intuitively says that at all times and for any fact φ (i.e., φ is a formula in our logic), if L knows φ, then L is permitted to know φ. As mentioned above, this intuitive explication of security was first suggested by Glasgow,

MacEwen, and Panangaden [GMP90] and further refined by Bieber and Cuppens [BC92]. We state SSC in our formalism as follows.

Definition 6.1 Let $L \subseteq C$ be a subject. Suppose a system Σ is described by a set of formulae in our logic, Γ. We say that Γ *satisfies the Syntactic Security Condition* (SSC) *with respect to* L if and only if for any formula φ,

$$\Gamma \vdash \Box(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi))$$

□

It is illuminating to consider for what kinds of formulae, φ, the sentence $\mathbf{K}_L(\varphi)$ is derivable but the formula $R_L(\varphi)$ is not (i.e., what kind of formulae distinguish secure systems from insecure ones). There are two ways in which this might occur. First, we may be able to derive the formula $\mathbf{K}_L(\varphi)$ from the set of premises Γ and the standard **S5** axioms for the $\mathbf{K}_L$ operator but not be able to derive $R_L(\varphi)$ from Γ and the standard **S5** axioms for R_L. Since the **S5** axioms are the same for $\mathbf{K}_L$ and for R_L, this would mean that the premises fairly directly imply that L knows φ but L is not permitted to know φ. However, in what we envision as the typical application of our logic, the set of premises, Γ, consists of a set of formulae saying that subjects always know that the system description always holds—where 'know' refers to weak-adversary knowledge. Given the axioms of our logic, from Γ we can also derive the set of formulae actually describing the system and the various other relevant temporal and epistemic formulae concerning the system description itself. Therefore, the formula $\mathbf{K}_L(\varphi)$ will be derivable from Γ and the standard **S5** axioms only in the case that $R_L(\varphi)$ is derivable from Γ and the standard **S5** axioms. Hence, we do not expect that the premises and the standard **S5** axioms alone will determine whether or not a system is secure.

The second way in which the formula $\mathbf{K}_L(\varphi)$ may be derived but not the formula $R_L(\varphi)$ is by using axiom **KO** (in conjunction with the other axioms, rules, and premises). Intuitively, axiom **KO** says that subjects always know the (conditional) distribution on the outputs that they can see. Recall that there is no corresponding axiom **RO**. Thus, subjects always *know* the (conditional) distribution on the outputs that they can see, but it is not necessarily the case that they are *permitted* to know that distribution. This is the essential difference between the two operators. And further, understanding this difference illuminates the nature of proving SSC; that is, proving SSC (with respect to some subject L) requires a proof

that L is permitted to know the (conditional) distribution on outputs to L. This would typically involve showing that this (conditional) distribution is logically derivable from other facts that L is permitted to know. In the typical application, these "other facts" would be the (conditional) distribution on inputs from L and the system description. Therefore, in the typical application, a system satisfies SSC (with respect to some subject L) only if the (conditional) distribution on outputs to L is logically derivable from the (conditional) distribution on inputs from L and the system description. As will be seen in §7, this point is important for practical verification purposes.

6.2 Relationship to Probabilistic Noninterference

In this subsection, we recall the definition of Probabilistic Noninterference (PNI) and prove that the semantic interpretation of SSC is equivalent to PNI. First, let's state the semantic interpretation of SSC.

Definition 6.2 Let $L \subseteq C$ be a subject. Suppose a system Σ is described by a set of formulae in our logic, Γ. We say that Γ *satisfies the Semantic Interpretation of the SSC with respect to* L if and only if for any formula φ,

$$\Gamma \models \Box(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi))$$

□

Now, we state the definition of PNI in terms of our model.

Definition 6.3 Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two adversaries that satisfy the Secure Environment Assumption. We will say that $\mathcal{A}_1$ and $\mathcal{A}_2$ *agree on* L *behavior* iff there exist $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{L}$ such that $\mathcal{H}_1$ and $\mathcal{L}$ are the unique probability functions that describe $\mathcal{A}_1$ (as in Definition 2.2) and $\mathcal{H}_2$ and $\mathcal{L}$ are the unique probability functions that describe $\mathcal{A}_2$. □

Observation 6.4 If $T_{\mathcal{A}_1}$ and $T_{\mathcal{A}_2}$ are Δ_L-accessible, then $\mathcal{A}_1$ and $\mathcal{A}_2$ agree on L behavior. □

Definition 6.5 Let Σ be a system with computation trees $\mathcal{T}(\Sigma)$. We say that Σ *satisfies Probabilistic Noninterference (PNI) with respect to a subject* $L \subseteq C$ iff for any two trees

satisfying the Secure Environment Assumption, $T_{\mathcal{A}}, T_{\mathcal{A}'} \in \mathcal{T}(\Sigma)$ and any L-event, e, if $\mathcal{A}$ and $\mathcal{A}'$ agree on L behavior, then

$$\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}'}(e)$$

□

PNI is equivalent to Browne's (independently developed) *Stochastic Non-Interference* [Bro89]. The significance of PNI is that it is arguably a necessary and sufficient condition for a system to be free of covert channels (cf. [Bro91]).

Before we prove the main result of this section, we state and prove a lemma.

Lemma 6.6 Suppose that $T_{\mathcal{A}}$ and $T_{\mathcal{A}'}$ are two trees that agree on L behavior (and satisfy the Secure Environment Assumption). Further suppose that for any two points, $P_1 \in T_{\mathcal{A}}$, $P_2 \in T_{\mathcal{A}'}$, and any low output vector, $b \in O[L]$, if P_1 and P_2 have the same L-history, then

$$v_{P_1}(Pr_L(L'_{out} = b)) = v_{P_2}(Pr_L(L'_{out} = b))$$

Then, for any L-event, e,

$$\mu_{\mathcal{A}}(e_{\mathcal{A}}) = \mu_{\mathcal{A}'}(e_{\mathcal{A}'})$$

Proof: First we prove this lemma for a certain subset of L-events, namely those L-events corresponding to a finite L-history.

Let e be an L-event such that there exists a time, k, (the length of e) and a characteristic run, ρ, such that for any run, ρ', $\rho' \in e$ iff ρ' has the same L-history as ρ up to time k. That is, e corresponds to the finite L-history characterized by ρ up to time k.

We now prove the lemma (for this subclass of L-events) by induction on the length of e.

Base case: The length of e is zero.

Since all runs have the same L-history up to time 0, the only two L-events of length 0 are the empty set, $\emptyset$, and the set of all runs from all trees, $\mathcal{R}$. In the former case,

$$\mu_{\mathcal{A}}(\emptyset_{\mathcal{A}}) = 0 = \mu_{\mathcal{A}'}(\emptyset_{\mathcal{A}'})$$

and in the latter case,

$$\mu_{\mathcal{A}}(\mathcal{R}_{\mathcal{A}}) = 1 = \mu_{\mathcal{A}'}(\mathcal{R}_{\mathcal{A}'})$$

Thus, the base case is proved.

Induction case: Assume the lemma holds for all L-events (corresponding to finite L-histories) of length k. Let e be an L-event corresponding to a finite L-history of length $k+1$. Suppose that ρ is a run that (up to time $k+1$) characterizes e.

Now, let e' be the L-event characterized by ρ up to time k. Intuitively, e' corresponds to the finite L-history obtained by truncating e at time k. By the induction hypothesis,

$$\mu_{\mathcal{A}}(e'_{\mathcal{A}}) = \mu_{\mathcal{A}'}(e'_{\mathcal{A}'}) \tag{2}$$

We have two cases.

Case 1: $\mu_{\mathcal{A}}(e'_{\mathcal{A}}) = 0$.

Note that $e \subseteq e'$. That is, every run that has the same L-history as ρ up to time $k+1$ also has the same L-history as ρ up to time k. Thus,

$$\mu_{\mathcal{A}}(e) \leq \mu_{\mathcal{A}}(e') \quad \text{and} \quad \mu_{\mathcal{A}'}(e) \leq \mu_{\mathcal{A}'}(e')$$

Further, since no event can have a negative measure, making use of Equation 2 we have that

$$\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}}(e') = \mu_{\mathcal{A}'}(e') = \mu_{\mathcal{A}'}(e) \tag{3}$$

Case 2: $\mu_{\mathcal{A}}(e'_{\mathcal{A}}) > 0$.

By Equation 2, we also have that $\mu_{\mathcal{A}'}(e'_{\mathcal{A}}) > 0$. Thus, by the definition of conditional probability,

$$\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}}(e') \cdot \mu_{\mathcal{A}}(e \mid e') \tag{4}$$

and

$$\mu_{\mathcal{A}'}(e) = \mu_{\mathcal{A}'}(e') \cdot \mu_{\mathcal{A}'}(e \mid e') \tag{5}$$

Let $\alpha \in I_{L,k}$ and $\beta \in O_{L,k}$ be the low input and output history, resp., that characterize e' and let $a \in I[L]$ and $b \in O[L]$ be the low input and output vectors at time $k+1$ that are needed to additionally characterize e. Then, by our construction of the probability measures $(\mu_{\mathcal{A}})$ and by the Secure Environment Assumption, we have that

$$\mu_{\mathcal{A}}(e \mid e') = \mu_{\mathcal{A}}(b, \mid e') \cdot \mathcal{L}(a \mid \alpha, \beta, k) \tag{6}$$

and

$$\mu_{\mathcal{A}'}(e \mid e') = \mu_{\mathcal{A}'}(b, \mid e') \cdot \mathcal{L}'(a \mid \alpha, \beta, k) \tag{7}$$

where $\mathcal{L}$ is the low environment of $\mathcal{A}$ and $\mathcal{L}'$ is the low environment of $\mathcal{A}'$.

Since $\mathcal{A}$ and $\mathcal{A}'$ agree on L-behavior,

$$\mathcal{L}(a \mid \alpha, \beta) = \mathcal{L}'(a \mid \alpha, \beta, k) \tag{8}$$

Further, since $\mu_{\mathcal{A}}(e') > 0$ and $\mu_{\mathcal{A}'}(e') > 0$, there exists points in both trees, $P_1 \in T_{\mathcal{A}}$ and $P_2 \in T_{\mathcal{A}'}$, each of whose L-histories are (α, β). By the assumptions in the lemma,

$$v_{P_1}(Pr_L(L'_{out} = b)) = v_{P_2}(Pr_L(L'_{out} = b))$$

But notice that $\mathcal{S}_{L,P_1} = e' = \mathcal{S}_{L,P_2}$. Therefore, by our definition of the assignment function,

$$\mu_{\mathcal{A}}(b \mid e') = \mu_{\mathcal{A}'}(b \mid e') \tag{9}$$

Thus, by Equations 6, 7, 8, and 9, we have that

$$\mu_{\mathcal{A}}(e \mid e') = \mu_{\mathcal{A}'}(e \mid e') \tag{10}$$

and finally, by Equations 2, 4, 5, and 10, we have that

$$\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}'}(e)$$

Thus, in both cases $\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}'}(e)$ and the induction case is proved.

Now, we can complete the proof by observing that every L-event can be constructed by taking a finite number of unions and complementations of L-events that correspond to finite L-histories. That is, the L-events that correspond to finite L-histories are analogous to the *generators* of our event spaces. Thus, the desired result that for an arbitrary L-event, e, $\mu_{\mathcal{A}}(e) = \mu_{\mathcal{A}'}(e)$ follows from the fact that the measures are equal on all of the L-events, $\{e_i\}$, that are used to construct e in this fashion. □

We can now prove the following theorem relating PNI and SSC.

Theorem 6.7 Let Γ be a set of formulae describing Σ and let $L \subseteq C$ be a subject. Then, Σ satisfies PNI with respect to L iff Γ satisfies the semantic interpretation of SSC with respect to L.

Proof: First we show the forward direction. Suppose Σ satisfies PNI and let P_1 be a point such that $P_1 \models \Gamma$. We must show that for any formula φ,

$$v_{P_1}(\Box(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi))) = \textbf{true} \tag{11}$$

Applying the semantic assignment function, v_{P_1}, to Formula 11, we get

$$\begin{aligned} &\text{For any point, } P_2, \text{ reachable from } P_1, \\ &\quad \text{if for any point } P_3, \kappa_L^{powerful}(P_2, P_3) \text{ implies } v_{P_3}(\varphi) = \textbf{true} \\ &\quad \text{then for any point } P_3, \delta_L(P_2, P_3) \text{ implies } v_{P_3}(\varphi) = \textbf{true} \end{aligned} \tag{12}$$

Let P_2 be a point reachable from P_1 and assume that

$$\text{for any point } P_3, \kappa_L^{powerful}(P_2, P_3) \text{ implies } v_{P_3}(\varphi) = \textbf{true} \tag{13}$$

Now, let P_3 be an arbitrary point. To prove Formula 11, it is sufficient to show that

$$\delta_L(P_2, P_3) \text{ implies } v_{P_3}(\varphi) = \textbf{true} \tag{14}$$

If $\kappa_L^{powerful}(P_2, P_3)$, then by Formula 13, $v_{P_3}(\varphi) =$ **true** and so Formula 14 holds. Therefore, assume that **not** $\kappa_L^{powerful}(P_2, P_3)$; that is, assume that either

1. P_2 and P_3 do not have the same L-history; or
2. for some L-event e, $\mu_{\mathcal{A}(P_2)}(e|\mathcal{S}_{L,P_2}) \neq \mu_{\mathcal{A}(P_3)}(e|\mathcal{S}_{L,P_3})$ (where $\mathcal{A}(P_2)$ is the adversary corresponding to the tree containing P_2 and $\mathcal{A}(P_3)$ is the adversary corresponding to the tree containing P_3).

Assuming that P_2 and P_3 do not have the same L-history (i.e., item 1 above), by the definition of δ_L we have **not** $\delta_L(P_2, P_3)$ and so Formula 14 is true. Therefore, assume that P_2 and P_3 *do* have the same L-history, but that item 2 holds. Let e be an L-event for which item 2 holds. We have two cases.

1. $\mu_{\mathcal{A}(P_2)}(e) \neq \mu_{\mathcal{A}(P_3)}(e)$. Since e is an L-event, PNI implies that $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ differ on L behavior.
2. $\mu_{\mathcal{A}(P_2)}(\mathcal{S}_{L,P_2}) \neq \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3})$. Since, by assumption, P_2 and P_3 have the same L-history, $\mathcal{S}_{L,P_2}$ and $\mathcal{S}_{L,P_3}$ represent projections of the same L-event onto their respective computation trees. Let e' be that L-event. Therefore, in this case, $\mu_{\mathcal{A}(P_2)}(e') \neq \mu_{\mathcal{A}(P_3)}(e')$ and, again, PNI implies that $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ differ on L behavior.

Thus, in either case, $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ differ on L behavior and therefore, by Observation 6.4, $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ cannot be Δ_L-accessible. Further, by the definition of δ_L we have **not** $\delta_L(P_2, P_3)$. Hence, Formula 14 is true.

Therefore, Formula 14 is true in all cases and Γ satisfies the semantic interpretation of SSC.

Now we show that if Γ satisfies the semantic interpretation of SSC (with respect to L), then Σ satifies PNI (with respect to L). Suppose that Σ does not satisfy PNI; that is, there exist adversaries, $\mathcal{A}$ and $\mathcal{A}'$, that satisfy the Secure Environment Assumption and that agree on L behavior, and an L-event, e, such that

$$\mu_{\mathcal{A}}(e) \neq \mu_{\mathcal{A}'}(e)$$

We want to show that Γ does not satisfy the semantic interpretation of SSC. To do so, it is sufficient to exhibit a point, P and a formula φ such that $P \models \Gamma$ and $P \not\models \Box(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi))$. We choose P and φ as follows.

Since $\mathcal{A}$ and $\mathcal{A}'$ agree on L behavior and there exists an L-event, e, such that

$$\mu_{\mathcal{A}}(e) \neq \mu_{\mathcal{A}'}(e)$$

by Lemma 6.6, there must exist two points, $P_1 \in T_{\mathcal{A}}$, $P_2 \in T_{\mathcal{A}'}$, and a low output vector, $b \in O[L]$, such that P_1 and P_2 have the same L-history and

$$v_{P_1}(Pr_L(L'_{out} = b)) \neq v_{P_2}(Pr_L(L'_{out} = b))$$

Let P_1 and P_2 be such points and suppose that, in fact,

$$v_{P_1}(Pr_L(L'_{out} = b)) = r \neq v_{P_2}(Pr_L(L'_{out} = b))$$

Therefore, choose $P = P_1$ and choose $\varphi = Pr_L(L'_{out} = b) = r$.

By axiom **KO**, and the soundness of our logic, $v_P(\mathbf{K}_L(\varphi)) =$ **true** . But, we have that $\delta_L(P, P_2)$ and $v_{P_2}(\varphi) =$ **false** . Therefore, $v_P(R_L(\varphi)) =$ **false** , and hence $v_P(\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)) =$ **false** .

Since Γ specifies Σ and $T_{\mathcal{A}}$ is a computation tree for Σ, $P \models \Gamma$ and the theorem is proved. □

The significance of this theorem is that (given soundness) verifying that a system satisfies SSC is sufficient to show that it satisfies PNI, which (as was previously mentioned) is a necessary and sufficient condition for a system to be free of covert channels. In the next section, we discuss the issue of verifying SSC.

7 Verification

Thus far in the paper, we have given a logic that can be used to specify a computer system and verify that it satifies PNI. This process consists of two steps: (1) specify the system under consideration as a set of premises Γ. (2) prove that $\Gamma \vdash \mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)$ for every formula φ.[6]

Since we do not quantify over formulae, it is impossible to formally deduce that for every formula φ, $\Gamma \vdash \mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)$ as this would require an infinite number of deductions.[7] Perhaps this shows that the verification effort is not pointed in the right direction. After all, many of formulae of the language, e.g., $2 + 2 = 4$, will have nothing to do with the security of a given system.

It thus seems desirable to find a verification condition that (1) is entirely expressible within our logic (i.e., it does not require metalinguistic variables such as φ), and (2) does not require the verifiers to prove things that have nothing to do with security. In the following two subsections, we give such a condition and discuss its relationship to previous work.

7.1 Syntactic Statement

In [McL90], McLean defines the Flow Model (FM) with the motivation of providing an abstract, but precise, explication of information flow security. McLean's intent for FM is to provide a characterization of security against which more concrete security models can be evaluated. In [Gra92], the first author studies a more concrete version of FM, called the Applied Flow Model (AFM), and it is shown therein that AFM captures a strictly stronger notion of security than PNI.

In this paper, we have another reason for studying AFM: it is more easily verified than SSC. It was already discussed above that proving SSC requires a proof that for any φ, the formula $\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)$ can be derived from the set of premises, Γ. The usual technique for such a proof is to proceed by induction on the structure of φ. (For example, one case of such a proof would be where φ is of the form $\psi \wedge \psi'$, and where the inductive hypothesis

[6] Actually, this would be done for each security class c by partitioning the set of communication channels into those that are dominated by c (which are called L) and those that are not dominated by c (which are called H).

[7] We can of course give an informal inductive proof on the structure of φ. But, this would not be a proof *in the logic*.

allows us to assume that both $\mathbf{K}_L(\psi) \rightarrow R_L(\psi)$ and $\mathbf{K}_L(\psi') \rightarrow R_L(\psi')$ are derivable from Γ.) Such a proof requires the prover to consider one case for each way that a formula φ can be constructed, and as noted above, many of these cases may have nothing to do with the security of the system under consideration.

As discussed in §6.1, the crucial difference between the **K** and R operators is that there is no axiom for R that corresponds to axiom **KO**. In particular, it is always the case that

$$Pr_S(c'_{out} = o) = r \rightarrow \mathbf{K}_S(Pr_S(c'_{out} = o) = r)$$

(for any given $S \subseteq C$, $c \in S$, $b \in O$, and $r \in \mathbb{R}$) but it is *not necessarily* the case that

$$Pr_S(c'_{out} = o) = r \rightarrow R_S(Pr_S(c'_{out} = o) = r)$$

Thus, if we can give a condition (i.e., a formula in our logic) that is sufficient to ensure that

$$Pr_L(c'_{out} = o) = r \rightarrow R_L(Pr_L(c'_{out} = o) = r)$$

is derivable from a set of premises Γ, then our intuition suggests that such a condition would be sufficient to ensure that $\mathbf{K}_L(\varphi) \rightarrow R_L(\varphi)$ is derivable from Γ, for any formula φ. The following definition provides such a condition.

Definition 7.1 Let $L \subseteq C$ be a subject. Suppose Γ is a set of premises that describe a system Σ. We say that Γ *satisfies the Syntactic Verification Condition* (SVC) *with respect to* L if and only if, for every $b \in O[L]$, the formula

$$\Box(Pr_C(L' = b) = r \rightarrow k_L(Pr_L(L' = b) = r))$$

is derivable from Γ. □

Intuitively, SVC says that at all times, assuming that the low environment is a weak adversary, he still knows the probability distribution on his next output.

In the next section, we will show that this statement is equivalent to a statement about conditional statistical independence. Namely, conditioned on the previous L-history, the next output on L is statistically independent of the previous non-L (i.e., high) history.

7.2 Relationship to Previous Formulations

In this section we show that $\Gamma \models$ SVC if and only if the system specified by Γ satisfies AFM (i.e., the relationship between SVC and AFM is analogous to the relationship between SSC and PNI).

Definition 7.2 Let Σ be a system with computation trees $\mathcal{T}(\Sigma)$ and let $L \subseteq C$ be a subject. We will say that Σ satisfies the *Applied Flow Model* (AFM) with respect to L iff for any tree, $T_{\mathcal{A}} \in \mathcal{T}(\Sigma)$ (satisfying the Secure Environment Assumption with respect to L), any point $P \in T_{\mathcal{A}}$, and any low output vector, $b \in O[L]$,

$$\mu_{\mathcal{A}}(\mathcal{S}_{C,P}(L' = b)) \;=\; \mu_{\mathcal{A}}(\mathcal{S}_{L,P}(L' = b))$$

□

This definition is, except for minor notational differences, exactly the definition of AFM as given in [Gra92]. Now we can prove the following theorem.

Theorem 7.3 Let Γ be a set of formulae describing Σ and let $L \subseteq C$ be a subject. Then, Σ satisfies AFM with respect to L iff Γ satisfies the semantic interpretation of SVC with respect to L.

Proof: Let $\mathcal{T}(\Sigma)$ be the set of computation trees for Σ. Suppose that Γ satisfies the semantic interpretation of SVC with respect to L. That is, for any point P_1 in any tree in $\mathcal{T}(\Sigma)$,

$$v_{P_1}\left(\Box(Pr_C(L' = b) = r \rightarrow k_L(Pr_L(L' = b) = r))\right) = \textbf{true}$$

By applying the semantic assignment function, we have for any point $P_2 \in \rho \in \mathit{extensions}(P_1)$ that

$$v_{P_2}(Pr_C(L' = b) = r) \Rightarrow v_{P_2}(k_L(Pr_L(L' = b) = r)))$$

Applying the semantic assignment function again, we have,

$$\mu_{\mathcal{A}(P_2)}(\mathcal{S}_{C,P_2}(L' = b)) = r \Rightarrow (\forall P_3, \kappa_L^{weak}(P_2, P_3) \Rightarrow \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3}(L' = b)) = r)$$

(where $\mathcal{A}(P_3)$ is the adversary corresponding to the tree containing P_3), which is equivalent to

$$\forall P_2, P_3 \left[\kappa_L^{weak}(P_2, P_3) \Rightarrow \mu_{\mathcal{A}(P_2)}(\mathcal{S}_{C,P_2}(L' = b)) = \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3}(L' = b))\right] \qquad (15)$$

Thus, Formula 15 is equivalent to the statement that Γ satisfies the semantic interpretation of SVC with respect to L. By choosing $P_2 = P_3 = P$ and by the reflexivity of κ_L^{weak}, Formula 15 implies that Σ satisfies AFM with respect to L.

We will now show that if Σ satisfies AFM with respect to L, then Formula 15 holds.

Suppose, for *reductio*, that $\kappa_L^{weak}(P_2, P_3)$, but $\mu_{\mathcal{A}(P_2)}(\mathcal{S}_{C,P_2}(L' = b)) \neq \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3}(L' = b))$.

Since $T_{\mathcal{A}(P_2)}, T_{\mathcal{A}(P_3)} \in \mathcal{T}(\Sigma)$, we may apply AFM (wrt L) to conclude that

$$\mu_{\mathcal{A}(P_2)}(\mathcal{S}_{L,P_2}(L' = b)) \neq \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3}(L' = b))$$

Recall from [Gra92] that any system satisfying AFM satisfies PNI (wrt the same subject). We will now show that the above equation is inconsistent with PNI, hence with AFM.

Suppose that $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ agree on low behavior. Then PNI is contradicted since $\mathcal{S}_{L,P}(L' = b)$ is the *tree*(P)-projection of an L-event for any point P. So, suppose that $T_{\mathcal{A}(P_2)}$ and $T_{\mathcal{A}(P_3)}$ disagree on low behavior. By the secure environment assumption, $\mathcal{A}(P_2)$ and $\mathcal{A}(P_3)$ can be given by

$$\begin{aligned}
\mathcal{A}(P_2)(a \mid \alpha, \beta, k) &= \mathcal{H}_2(a\restriction(C - L) \mid \alpha, \beta, k) \cdot \mathcal{L}_2(a\restriction L \mid \alpha\restriction L, \beta\restriction L, k) \\
\mathcal{A}(P_3)(a \mid \alpha, \beta, k) &= \mathcal{H}_3(a\restriction(C - L) \mid \alpha, \beta, k) \cdot \mathcal{L}_3(a\restriction L \mid \alpha\restriction L, \beta\restriction L, k)
\end{aligned}$$

We can define a new adversary $\mathcal{A}_4$, which satisfies the Secure Environment Assumption, by

$$\mathcal{A}_4(a \mid \alpha, \beta, k) = \mathcal{H}_3(a\restriction(C - L) \mid \alpha, \beta, k) \cdot \mathcal{L}_2(a\restriction L \mid \alpha\restriction L, \beta\restriction L, k)$$

Thus, we also have $T_{\mathcal{A}_4} \in \mathcal{T}(\Sigma)$. We now show by induction on prefixes of the C-history of P_3 that $T_{\mathcal{A}_4}$ contains a point P_4 with the same history on all channels as P_3, i.e., such that $\kappa_C^{weak}(P_3, P_4)$. Obviously $T_{\mathcal{A}_4}$ contains the empty trace. Suppose that the time of P_3 is k and that there is a point in $T_{\mathcal{A}_4}$ with the same C-history as P_3 through time $k' < k$. By construction of the computation trees, and since $\kappa_L^{weak}(P_2, P_3)$, the input and output vectors that extend the subhistory of P_3 to $k' + 1$ are assigned a positive branch probability in $T_{\mathcal{A}_4}$. Therefore, by the structure of trees, there is a point $P_4 \in T_{\mathcal{A}_4}$ with the same C-history as P_3 through time $k' + 1$.

Thus, by construction, $\mathcal{A}(P_2)$ and $\mathcal{A}(P_4)$ agree on L behavior, but

$$\mu_{\mathcal{A}(P_4)}(\mathcal{S}_{L,P_4}(L' = b)) = \mu_{\mathcal{A}(P_3)}(\mathcal{S}_{L,P_3}(L' = b)) \neq \mu_{\mathcal{A}(P_2)}(\mathcal{S}_{L,P_2}(L' = b))$$

However, this contradicts PNI, hence AFM, and our supposition is discharged.

□

Since, as remarked, AFM is stronger than PNI [Gra92], the foregoing theorem shows that SVC is a sufficient condition for a system to satisfy PNI.

7.3 Examples, continued

We note here that the security of the encryption box of Example 3.1 with respect to a subject $L \subseteq C$ is formally derivable. In fact, once the assumptions are written down, there is virtually nothing to prove. Recall the system specification: If $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$, then, the system is specified by the following formula.

$$\Box\,(Pr_C(l'_{out} = 0) = Pr_C(l'_{out} = 1) = 0.5)$$

(In the initial specification relativisation to C was left implicit for simplicity since it is tantamount to relativising to the system, Σ.) Recall also that subjects are assumed to always know that the system description holds at all times. Thus,

$$\Gamma = \{\Box k_L \Box\,(Pr_L(L'_{out} = 0) = Pr_L(L'_{out} = 1) = 0.5)\}$$

The only $b \in O[L]$ are O and 1; hence, the only antecedents for the SVC schema that are consistent with Γ are $Pr_C(L'_{out} = 0)$ and $Pr_C(L'_{out} = 1)$. Thus, SVC with respect to L for this system is:

$$\Box\,(Pr_C(L'_{out} = 0) = 0.5 \wedge Pr_C(L'_{out} = 1) = 0.5) \rightarrow k_L\,(Pr_L(L'_{out} = 0) = 0.5 \wedge Pr_L(L'_{out} = 1) = 0.5)$$

But, this is obviously derivable from Γ.

We also observe that for the insecure encryption box of Example 3.2 $\Gamma \nvdash SSC$ (where Γ encompasses those formulae that embody the system description and our assumptions about knowledge thereof). It is obvious that the insecure encryption box fails to satisfy PNI. By the attack described in the original example, we can easily find two adversaries that satisfy the Secure Environment Assumption and agree on low behavior and yet disagree on the probability of certain low events. Indeed, the low environment can assign 0/1 probabilities

to any output sent by the high part of the adversary. By theorem 6.7, we thus have that $\Gamma \not\models SSC$. And, by soundness (theorem 5.1), it follows that $\Gamma \not\vdash SSC$.

8 Conclusions

We have given a logic for specifying and reasoning about the multilevel security of probabilistic computer systems. Beside the practical benefits of providing a logic to reason about probabilistic systems, we have established connections between previous logical formulations of security (viz, [GMP90] and [BC92]), information-theoretic formulations of security (viz, [Bro89] and [Gra92]), and logical formulations of knowledge and probability in distributed systems (viz, [HT93]). These connections serve to increase our confidence that each formulation is correct.

References

[Aba90] Martín Abadi. An axiomatization of Lamport's temporal logic of actions. Technical Report 65, SRC, October 1990.

[BC92] P. Bieber and F. Cuppens. A logical view of secure dependencies. *Journal of Computer Security*, 1(1):99–129, 1992.

[Bro89] Randy Browne. *Stochastic Non-Interference: Temporal Stochastic Processes Without Covert Channels.* Odyssey Research Associates, Ithaca, NY, November 1989. unpublished manuscript.

[Bro91] Randy Browne. The Turing Test and non-information flow. In *Proc. 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1991.

[Che80] Brian F. Chellas. *Modal Logic: An Introduction.* Cambridge University Press, Cambridge, 1980.

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *JACM*, 40(1):17–47, January 1993.

[Den76] Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[Den82] Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley, Reading, Massachusetts, 1982.

[FH94] Ronald Fagin and Joseph Y. Halpern. Reasoning about knowledge and probability. *JACM*, 41(2):340–367, March 1994.

[Gas88] Morrie Gasser. *Building a Secure Computer System.* Van Nostrand Reinhold, New York, New York, 1988.

[GMP90] Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. In *Proc. Computer Security Foundations Workshop III*, Franconia, NH, June 1990.

[Gol92] Robert Goldblatt. *Logics of Time and Computation, 2^{nd} edition*, volume 7 of *CSLI Lecture Notes.* CSLI Publications, Stanford, California, 1992.

[Gra92] James W. Gray, III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3):255–294, 1992. A preliminary version appears in Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.

[GS92] James W. Gray, III and Paul F. Syverson. A logical approach to multilevel security of probabilistic systems. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 164–176, Oakland, CA, May 1992.

[Hal50] Paul R. Halmos. *Measure Theory.* Springer-Verlag, New York, New York, 1950.

[HT93] Joseph Y. Halpern and Mark R. Tuttle. Knowledge, probability, and adversaries. *JACM*, 40(4):917–962, September 1993.

[Lam73] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10), October 1973.

[Lam91] Leslie Lamport. The temporal logic of actions. Technical Report 79, DEC Systems Research Center, Palo Alto, CA, December 1991.

[Mar74] Jerrold E. Marsden. *Elementary Classical Analysis.* W.H. Freeman, San Francisco, 1974.

[McC88] Daryl McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, 1988.

[McC90] Daryl McCullough. A hookup theorem for multilevel security. *IEEE Transactions on Software Engineering*, 16(6):563–568, June 1990.

[McL87] John McLean. Reasoning about security models. In *Proc. 1987 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1987.

[McL90] John McLean. Security models and information flow. In *Proc. 1990 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 1990.

[Mea92] Catherine Meadows. Applying formal methods to the analysis of a key management protocols. *Journal of Computer Security*, 1(1):5–35, 1992.

[Mil90] Jonathan K. Millen. Hookup security for synchronous machines. In *Proceedings of the Computer Security Foundations Workshop III*, Franconia, NH, June 1990.

[MR88] Leo Marcus and Timothy Redmond. A model-theoretic approach to specifying, verifying, and hooking up security policies. In *Proceeding of the Computer Security Foundations Workshop*, Franconia, NH, 1988.

[Rud] Walter Rudin. *Principals of Mathematical Analysis.* McGraw-Hill, New York.

[Rus87] Enrique H. Ruspini. Epistemic logics, probability, and the calculus of evidence. In *Proceedings of the* 10^{th} *International Joint Conference on Artificial Intelligence*, pages 924–931, 1987.

[Sei92] Karen Seidel. Probabilistic communicating processes. Technical report, University of Oxford, Lady Margaret Hall, 1992. PhD thesis.

[Sha58] Claude E. Shannon. Channels with side information at the transmitter. *IBM Journal of Research and Development*, 2:289–293, October 1958. Republished in: David Slepian (ed.), "Key Papers in the Development of Information Theory", IEEE Press, 1974.

[Shi84] A. N. Shiryayev. *Probability*, volume 95 of *Graduate Tests in Mathematics.* Springer-Verlag, 1984.

[Sut86] David Sutherland. A model of information. In *Proceeding of the 9th National Computer Security Conference*, Baltimore, MD, September 1986.

[WJ90] J. Todd Wittbold and Dale M. Johnson. Information flow in nondeterministic systems. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, 1990.

[Wra92] John C. Wray. An analysis of covert timing channels. *The Journal of Computer Security*, 1(3):219–232, 1992.